균등 격자를 이용한 공간 클러스터링 기법의 설계 및 구현

문상호

부산외국어대학교 컴퓨터공학부

Design and Implementation of Spatial Clustering Method using Regular Grid

Sang-Ho Moon

Division of Computer Engineering, Pusan University of Foreign Studies

E-mail : shmoon87@pufs.ac.kr

요 약

기존 연구에서 공간데이타 마이닝을 지원하기 위하여 여러 가지 공간 클러스터링 기법들이 제시되었다. 그러나 대부분의 기법들이 객체들 간의 거리를 기반으로 수행하므로, 공간데이타의 양이 많아질수록 계산 비용이 증가하는 문제점이 발생한다. 본 논문에서는 이러한 문제점을 해결하기 위하여, 균등 격자를 기반으로 하는 공간 클러스터링 기법을 제시한다. 그리고 이 기법을 실현화시키기 위하여 파일구조, 자료구조, 알고리즘을 설계 및 구현하고, 실제 실험데이타를 대상으로 적용하여 클러스터 생성 결과를 보인다.

ABSTRACT

Several clustering methods for spatial data mining have been devised in the literature, but have the following drawback: increase cost due to calculating distance among objects. To solve this problem, we propose a spatial clustering method using regular cells. In this paper, we design and implement file structures, data structures and algorithms to realize the proposed method, also, show experimental results after applying test data to the implemented method.

키워드

공간 클러스터링, 공간데이타 마이닝, 균등 격자, 공간 데이타베이스

1. 서 론

공간데이타 마이닝은 공간 DB로부터 암시적이며 잠재적인 지식을 추출하는 과정이다[1,2,3,4]. 그리고 공간 클러스터링은 공간데이타 마이닝 기법중의 하나로, 공간객체들에 대하여 공간적 특성을 이용하여 집단화하는 과정이다. 이러한 공간 클러스터링은 다른 마이닝 알고리즘의 전처리 단계로 이용되거나 유사성 검색 등의 많은 응용 분야에 널리 사용되고 있다.

공간 DB는 다양하고 복잡한 공간데이타를 포함하므로, 암시적이고 잠재적인 유용한 지식을 발견하기 위해서는 많은 비용이 든다. 이로 인하여 전체 데이타로부터 의미 있는 부분집합을 찾아내고, 이 부분집합을 대상으로 유용한 지식을 추출하는 것이 필요하다. 따라서 공간데이타 마이닝에서 공간 클러스터링은 중요한 역할을 담당하며, 복잡도가 큰 탐색공간에서 효율적인 공간 클러스터링 기법의 개발이 요구된다.

기존 연구에서 많은 공간 클러스터링 기법들이 제시되었다. 그러나 대부분이 객체들의 거리 계산을 기반으로 하므로 데이타 양이 많아질수록 비용이 커진다. 또한, 메모리 상주 데이타를 대상으로 하므로 대용량의 데이타인 경우에 효율이 떨어진다. 본 논문에서는 이러한 문제점을 해결하기 위하여 균등 격자(regular grid)를 이용한 공간 클러스터링 기법을 제시한다. 이 기법에서는 방대한 양의 공간데이타를 대상으로 효율적인 클러스터링을 위하여 계산 비용 감소에 중점을 둔다. 세부적으로 객체간의 거리 계산을 대체하여 균등 격자의 관련성을 이용하여 클러스터링을 수행한다.

본 논문에서는 제시한 공간 클러스터링 기법을 구현하기 위하여, 먼저 파일구조와 자료구조를 설계한다. 그리고 격자파일을 이용한 공간 클러스터

링 알고리즘을 설계 및 구현한다. 마지막으로 실제 실험데이타를 대상으로 격자파일과 클러스터 생성 결과를 보인다.

II. 공간 클러스터링 기법

2.1 기존 클러스터링 기법

기존에 공간 클러스터링을 위하여 여러 기법들이 제시되었다. DBSCAN은 데이타 밀도를 기반으로 한 알고리즘이며[2], CLARANS는 데이타마이닝을 위해 제안된 PAM과 CLARA를 결합하여 제안한 알고리즘이다[1]. H-SCAN은 공간데이타를 클러스터링하기 위하여 1차원을 위한 해시 방법을 확장하여 d차원 공간에서 사용한 것이며[4], STING은 공간데이타 마이닝을 위하여 통계정보를 그리드 셀 계층구조를 관리하여 이용한다[3]. 그러나 이러한 기법들은 대부분이 메모리상의 데이터를 대상으로 거리를 기반으로 클러스터링을 수행하므로, 계산 비용이 많아지고 대용량의 데이타인 경우에 효율이 떨어지는 문제점이 발생한다.

2.2 균등 격자를 이용한 클러스터링 기법

본 논문에서 제시하는 공간 클러스터링 기법의 핵심은 거리 계산을 최소화하기 위하여 균등 격자를 기반으로 셀 관련성을 이용하여 클러스터링을 하는 것이다. 먼저 그림 1에서와 같이 전체 공간영역을 임계값을 기준으로 셀 크기를 결정하여 균등 격자구조를 생성한다. 여기서 *C66*을 기준으로 직접인접한 8개의 셀들(*DAC(C66)*)은 거리 계산없이 하나의 클러스터로 묶을 수 있다. 이것은 셀들 간의 가장 먼 거리가 대각선들의 합이므로 셀 내의 모든 객체들은 임계값 내에 있기 때문이다. 그리고 이 인접한 셀들 중에서 객체가 있는 셀을 대상으로 반복해서 직접인접 셀들을 찾으면, 클러스터를 생성할 수 있다. 그림 1에서는 결과적으로 2개의 클러스터들이 생성된다.

균등 격자를 이용하여 추출한 클러스터들은 최종 결과는 아니다. 왜냐하면 클러스터들에 속한 객체들의 거리를 비교하면 임계값 이내에 존재하는 객체들이 있을 수 있기 때문이다. 예를 들어, 그림 1에서 *Cluster1*과 *Cluster2*가 생성되었지만, *Cluster1*에 속하는 객체를 기준으로 보면 *Cluster2*의 일부 객체들이 임계값 이내에 포함됨을 알 수 있다. 따라서 이러한 경우를 고려하여 후보 클러스터들 간의 합병 여부를 판단해야 한다.

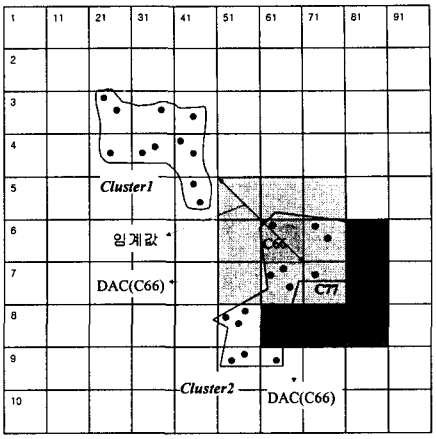


그림 1. 균등 격자를 이용한 클러스터링

III. 클러스터링 알고리즘

공간 클러스터링 기법을 구현하기 위하여 본 논문에서는 세부 알고리즘을 구현한다. 이 알고리즘과 관련하여 직접인접, 직접인접셀, 인접가능, 전파가능, 클러스터링 가능셀/범위에 대한 정의가 필요하다. 이 정의와 알고리즘에 대한 세부적인 내용은 [5]에 자세하게 기술되어 있다.

3.1 클러스터 생성 알고리즘

클러스터링 생성 알고리즘은 균등 격자구조에서 셀 관련성만으로 후보 클러스터를 생성한다. 이때, 직접인접, 직접인접 셀, 인접가능, 전파가능 정의를 이용한다. 세부적으로 이 알고리즘에서는 셀 관련성을 기반으로 전파(propagation) 방법을 이용하여 클러스터를 생성한다. 클러스터 생성 알고리즘은 그림 2와 같다.

Step 1) 격자구조에서 객체들이 존재하는 셀들을 검색
Step 2) 기준셀(C_i)을 정하고, 직접연결셀($DAC(C_i)$)을 검색
Step 3) 전파가능한 셀들을 찾기 위하여 2단계에서 검색된 셀들($C_j \in DAC(C_i)$)에 대하여 직접연결 셀($DAC(C_j)$)을 검색
Step 4) 셀들이 검색될 때까지 2, 3단계를 반복 수행
Step 5) 검색된 셀들을 묶어서 하나의 클러스터를 생성
Step 6) 전체셀에 대해 2~5단계를 반복 수행하여 후보 클러스터 찾음

그림 2. 클러스터 생성 알고리즘

3.2 클러스터 합병 알고리즘

전단계에서 후보 클러스터들은 셀 관련성만을 이용하여 생성되었으므로, 클러스터들 간의 합병 여부를 확인하는 과정이 필요하다. 이 과정에서 객체들 간의 거리 계산을 최소화하기 위하여 클러스터링 가능셀/범위 정의를 이용한다. 세부적으로 한 클러스터의 셀을 기준으로 하여 합병가능한 영역을 구한 후에, 이 영역에 속하는 셀들을 대상으로 거리 계산을 수행한다. 예를 들어, 그림 1에서 *Cluster1*의 *C45*와 클러스터링 가능셀인 *C66*에 속한 객체들 간의 거리 계산을 하면 임계값 이내에 있으므로 2개의 클러스터들은 하나로 합병이 된다. 그림 3은 클러스터 합병 알고리즘을 보여준다.

Step 1) 후보 클러스터들에 대하여 클러스터링 가능셀 범위(*ECCA*)를 구함
Step 2) 셀 범위내에서 클러스터링 가능셀(*ECC*)의 존재 여부 확인
Step 3) *ECC*가 존재하면, *ECC* 내부의 객체들간에 거리 계산
Step 4) 객체 거리가 임계값 이내에 있으면 후보 클러스터들을 합병
Step 5) 후보 클러스터들에 대하여 1~4 단계를 반복 수행

그림 3. 클러스터 합병 알고리즘

IV. 격자파일 설계

기존의 공간 클러스터링 기법은 메모리 상주 데이터만을 대상으로 하므로 대용량의 데이터인 경우에는 효율이 떨어지는 문제점이 발생한다. 본 논문에서는 이러한 문제점을 해결하기 위하여 모든 정보를 파일로 처리하도록 한다. 이를 위하여, 공간데이타를 저장하기 위한 데이터파일과 이 파일로부터 균등 격자구조를 생성하여 저장하는 격자파일(grid file) 구조를 설계한다.

4.1 데이터파일 구조

실험평가를 위하여 제공되는 대부분의 샘플데이타들은 텍스트 형식이다. 본 논문에서는 알고리즘의 효율성을 높이기 위하여 텍스트 형식을 이진(binary) 형식으로 변환하여 데이터 파일을 생성한다. 데이터 파일은 점 객체의 실제 위치를 나타내는 것으로, 객체의 식별자(OID)와 좌표값을 x, y의 형태로 저장한다. 데이터 파일의 구조는 그림 4와 같다.

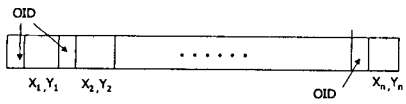


그림 4. 데이터파일 구조

4.2 격자파일 구조

격자파일은 전체 공간영역에 대하여 임계값에 따라 나누어진 셀들의 구조를 나타내며, 크게 셀의 기본 정보를 나타내는 셀-헤드와 셀에 속하는 객체들에 대한 정보를 나타내는 셀-디렉토리로 구성된다. 셀-헤드는 셀의 정보를 나타내주는 부분으로서, 전체 셀의 수만큼 저장된다. 내부적으로 셀-디렉토리에서 셀의 시작지점을 나타내는 offset, 셀 내의 객체수, 셀 영역으로 구성된다. 그림 5는 셀-헤드 구조를 나타낸다.

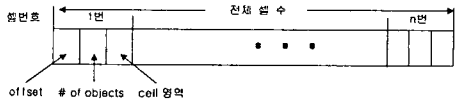


그림 5. 셀-헤드(cell-head) 구조

셀-디렉토리는 셀에 속하는 데이터에 대한 정보를 나타낸다. 세부적으로 셀에 속하는 공간객체들의 식별자들로 구성되며, 실제 객체들의 좌표는 데이터 파일에서 식별자를 이용하여 검색한다. 그림 6은 셀-헤드 구조를 나타낸다.

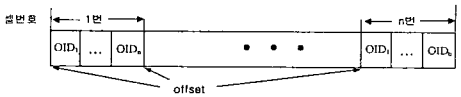


그림 6. 셀-디렉토리(cell-directory) 구조

4.2 출력파일 구조

출력파일은 격자파일과 데이터파일을 기반으로 클러스터링 알고리즘을 적용하여 생성된 클러스터를 저장하기 위한 것이다. 내부적으로 클러스터를 구별하는 flag와 각 클러스터는 포함되는 셀들의 번호를 저장하며, 이 셀 번호를 이용하여 실제 클러스터에 포함되는 객체들을 검색한다. 출력파일의 구조는 그림 7과 같다.

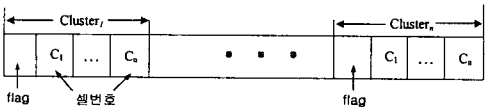


그림 7. 출력(output) 파일 구조

V. 구현 및 실험평가

5.1 실험데이타

실험데이타는 데이터 분포를 기준으로 2개의 DataSet을 이용하였으며, 실험의 공정성을 위하여 [14]의 Scholl Benchmark 데이타 집합으로부터 생성된 MBR 데이타의 최소 좌표점을 객체의 위치로 하는 데이타를 생성하여 이용하였다. 각 실험데이타에 대한 특성은 표 1과 같다.

표 1. 실험데이타의 특성

실험 데이타	범위 (X,Y축)	X축 분포	Y축 분포	객체수	임계값
DataSet1	0-10,000	gaussian	exponential	1000	500
DataSet2	0-10,000	exponential	gaussian	100	1000

5.2 격자파일 생성

데이터파일에서 격자파일로 생성하는 과정은 메모리상에서 수행된다. 이를 위하여 격자파일 생성에 필요한 자료구조를 먼저 정의한다. 그림 8은 자료구조에 대한 내용이다.

```
typedef struct _FileHeader{
      char layer_name[32];
      int t_value;
      int minx, miny, maxx, maxy;
  } FILE_HEADER;
typedef struct _Geom{
      int oid, approx_x, approx_y;
  } GEOM;
typedef struct _approx{
      int oid, offset;
  } APPROX;
typedef struct _cell{
      int offset, minx, miny, maxx, maxy;
      CArray<Approx,Approx> *apprs;
  } GRID_CELL;
```

그림 8. 격자파일 생성을 위한 자료구조

*FILE_HEADER*는 격자파일 생성에 기본이 되는 파일이름, 임계값, 전체 공간영역을 가진다. *GEOM*은 객체의 oid와 실제 좌표를, *APPROX*는 oid와 데이터파일 내의 위치를 나타내는 offset을 저장한다. *GRID_CELL*은 셀 정보를 나타내며 셀영역, 셀-디렉토리상의 위치, 셀내에 속하는 객체들을 저장한다. 이 구조체들을 이용하여 격자파일을 생성하는 과정은 그림 9와 같다. 여기서 *InitGridFile()* 함수는 임계값을 기준으로 전체 영역을 분할하여 셀들로 나누고, *Cell_Adjust()*는 셀내에 포함되는 객체 와 파일에 저장 위치를 결정하여 격자파일을 생성한다.

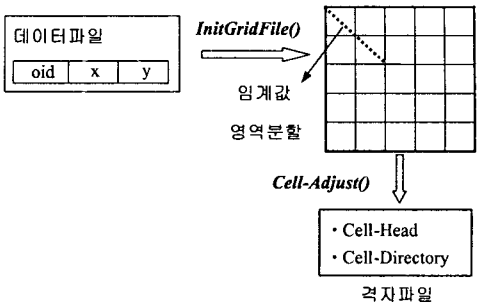


그림 9. 격자파일 생성 과정

그림 10은 실험 데이터파일을 이용하여 생성한 격자파일을 보여준다.

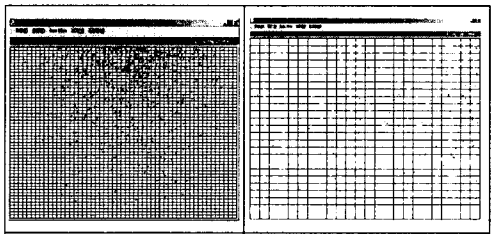


그림 10. 격자파일 출력

5.3 실험평가

그림 11은 클러스터 생성 알고리즘을 적용한 결과를 보여주며, 여기서 같은 색이나 번호를 가지는 셀들을 동일한 클러스터를 나타낸다. 그림 11(a)에서는 103개의 후보 클러스터들로 군집을 형성한다. 중앙 상단의 경우에는 크기가 큰 하나의 클러스터로 형성되었지만, 하단의 경우에는 크기가 작은 다수의 클러스터들 집합이 생성됨을 알 수 있다. 그림 11(b)에서는 31개의 클러스터들이 형성되었으며, 데이터의 분포에 따라 우측으로 클러스터들이 집중되었다.

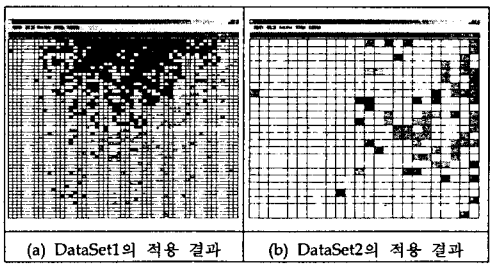
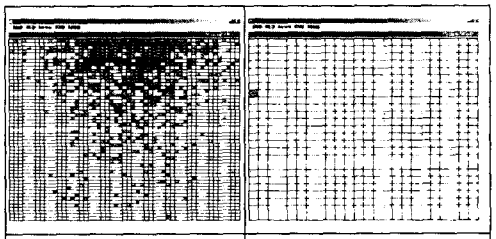


그림 11. 후보 클러스터 생성 결과

그림 12는 후보 클러스터들에 대하여 합병 알고리즘을 적용한 결과를 보여준다. 그림 12(a)에서는 103개의 후보 클러스터들을 형성하던 DataSet1이 12개의 클러스터들로 합병되었다. 그리고 그림 12(b)에서는 DataSet2의 후보 클러스터들에 대하여 합병 여부를 판단하여 합병한 클러스터들의 결과를 보여준다.



(a) DataSet1의 적용 결과 (b) DataSet2의 적용 결과

그림 12. 후보 클러스터들의 합병 결과

V. 결 론

본 논문에서는 균등 격자를 이용한 공간 클러스터링 기법을 제시하였다. 그리고 이 기법들을 실현하기 위하여, 파일구조와 자료구조를 정의하고, 격자파일을 이용한 공간 클러스터링 알고리즘을 설계 및 구현하였다. 마지막으로 실험데이타를 대상으로 실험결과를 보였다. 향후 연구로는 제안한 클러스터링 기법의 성능을 입증하기 위하여 다른 클러스터링 기법들과의 성능평가를 수행할 계획이다.

참고문헌

[1] Ng and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining", Int. Conf. on VLDB, pp.144~155, 1994.

[2] M. Ester, H.P. Kriegel, J. Sander, and X. Xu., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Int. Conf. on KDD, pp.226~231, 1996.

[3] W. Wang, J. Yang and R. Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining", Int'l Conf. on VLDB, pp.186-195, 1997.

[4] 오병우, 한기준, "H-SCAN: 지식 추출을 위한 해시-기반 공간 클러스터링 알고리즘", 한국정보과학회 논문지, 26권 7호, pp.857~869, 1999.

[5] 문상호, 이동규, 서영덕, "공간데이타 마이닝을 위한 효율적인 그리드 셀 기반 공간 클러스터링 알고리즘", 정보처리학회논문지, 10-D권, 4호, 2003.

[6] Spatial Join Benchmarking(http://www.enst.fr/~bdtest/sigbench/index.html)